CLAIMS

1. A method, implemented in a data processing system, for determining run-time dependencies between logical components of a data processing environment, the method comprising the steps of:

monitoring run-time activity of each of a first logical component and a second logical component of the data processing environment;

comparing the monitored run-time activity of the first logical component with the monitored run-time activity of the second logical component to identify correlations between the monitored run-time activity of the first and second logical components; and

in response to identification of a positive correlation between the monitored run-time activity of the first and second logical components, recording the existence of a dependency relationship between the first and second logical components.

2. The method of claim 1, further comprising the steps of:

monitoring run-time activity data of each of a plurality of logical components additional to said first and second logical components;

comparing the monitored run-time activity of the first logical component with the monitored run-time activity of the plurality of logical components to identify positive correlations between the monitored run-time activity of the first logical component and the monitored run-time activity of any of the plurality of logical components; and

recording the existence of a dependency relationship between the first logical component and any of the plurality of logical components for which a positive correlation is identified.

3. The method of claim 2, further comprising aggregating the recorded dependency relationships of the first logical component.

4. The method of claim 2, further comprising:

comparing the monitored run-time activity of the second logical component with the run-time activity of each of the plurality of logical components to identify positive

correlations between the monitored run-time activity of the second logical component and the monitored run-time activity of any of the plurality of logical components; and

recording the existence of a dependency relationship between the second logical component and any of the plurality of logical components for which a positive correlation is identified.

5. The method of claim 1, wherein the step of monitoring run-time activity comprises determining an activity period for each of the first and second logical components, and the comparing step comprises comparing the activity periods of the first and second logical components to identify correlations between the activity periods of the first and second logical components.

6. The method of claim 5, wherein the comparing step comprises determining whether the activity period of the first logical component contains the activity period of the second logical component.

7. The method of claim 1, wherein the step of monitoring run-time activity comprises monitoring invocations of the first and second logical components, and the comparing step comprises comparing the number of invocations of each of the first and second logical components within a monitoring period.

8. The method of claim 1, further comprising:

monitoring run-time activity for a plurality of executions of each of a first and a second logical component,

comparing the monitored run-time activity for the plurality of executions of the second logical component with the monitored run-time activity for the plurality of executions of the first logical component;

determining a proportion of executions of the first logical component for which a positive correlation is identified between the compared run-time activity of the first and second logical components; and

recording in association with the recorded dependency relationship a value

representing the determined proportion of executions of the first logical component for which a positive correlation is identified.

9. The method of claim 1, further comprising:

monitoring run-time activity for a plurality of executions of each of a first and a second logical component,

comparing the monitored run-time activity for the plurality of executions of the second logical component with the monitored run-time activity for the plurality of executions of the first logical component;

determining a proportion of executions of the second logical component for which a positive correlation is identified between the compared run-time activity of the first and second logical components; and

recording in association with the recorded dependency relationship a value representing the determined proportion of executions of the second logical component for which a positive correlation is identified.

10. The method of claim 8, further comprising:

determining a proportion of executions of the second logical component for which a positive correlation is identified between the compared run-time activity of the first and second logical components; and

recording in association with the recorded dependency relationship a value representing the determined proportion of executions of the second logical component for which a positive correlation is identified; and

generating a weight value comprising a combination function of the determined proportions of executions of the first and second logical components, and storing the weight value in association with the recorded dependency relationship.

11. The method of claim 1, wherein the step of monitoring the run-time activity of the first and second logical components comprises generating events in response to completion of the processing of requests by each of the first and second logical components, and wherein the step of comparing the monitored run-time activity

comprises the steps of:

calculating an activity period for each of the first and second logical components in response to generated events indicating the completion of processing of a request by the respective one of the first and second logical components; and

determining whether the activity period of the first component contains the activity period of the second component.

12. The method of claim 11, comprising:

computing a probability value associated with a dependency relationship by determining, for a plurality of executions of the first and second components, the proportion of executions of the first component for which the activity period of the first component contains the activity period of the second component, and recording the probability value in association with the recorded dependency relationship.

13. The method of claim 11, further comprising:

computing a probability associated with a dependency relationship by determining, for a plurality of executions of the first and second components, the proportion of executions of the second component for which the activity period of the first component contains the activity period of the second component, and recording the probability value in association with the recorded dependency relationship.

14. The method of claim 11, wherein the events are generated in response to a monitoring agent polling the first and second components for information relating to the processing of requests.

15. The method of claim 11, wherein the events are generated by a publish/subscribe broker which receives run-time activity data from the first and second logical components, identifies a subscriber by reference to stored subscription information, and sends the generated events to the identified subscriber.

16. The method of claim 1, for monitoring a data processing system which comprises

a monitoring interface for accessing run-time activity data within the data processing system, wherein the step of monitoring run-time activity comprises a monitoring agent accessing run-time activity data via the monitoring interface.

17. The method of claim 16, wherein the monitoring agent comprises an aggregator for aggregating run-time activity data accessed via the monitoring interface.

18. The method of claim 16, wherein the monitoring agent comprises program code for computing run-time activity metrics from the run–time activity data accessed via the monitoring interface.

19. The method of claim 16, wherein the monitoring agent is configured to poll the first and second logical components via the monitoring interface.

20. A method of fault management comprising the steps of:

monitoring run-time activity of each of a first logical component and a plurality of additional logical components of a data processing environment;

comparing the monitored run-time activity of the first logical component with the monitored run-time activity of each of the plurality of logical components, to identify positive correlations between the monitored run-time activity of the first logical component and the monitored run-time activity of any of the plurality of logical components;

recording the existence of a dependency relationship between the first logical component and any of the plurality of logical components for which a positive correlation is identified;

aggregating the recorded dependency relationships to determine a set of logical components having dependency relationships with the first logical component; and

responding to identification of a problem affecting the first logical component by analysing the set of logical components having dependency relationships with the first logical component.

21. The method of claim 20, wherein the step of aggregating the recorded dependency relationships comprises sorting the dependencies into an order determined by a sorting heuristic, and the step of analysing the set of logical components comprises analysing components of the set of components in said determined order.

22. A method for determining a set of logical components of a data processing environment which are likely to be affected by termination of a first logical component of the data processing environment, the method comprising:

monitoring run-time activity of each of a first logical component and a plurality of additional logical components of a data processing environment;

comparing the monitored run-time activity of the first logical component with the monitored run-time activity of each of the plurality of logical components, to identify positive correlations between the monitored run-time activity of the first logical component and the monitored run-time activity of any of the plurality of logical components; and

recording the existence of a dependency relationship between the first logical component and any of the plurality of logical components for which a positive correlation is identified; and

aggregating the recorded dependency relationships to determine a set of logical components having dependency relationships with the first logical component.

23. A computer program product comprising program code recorded on a recording medium, for controlling the operation of a data processing system on which the program code executes to determine run-time dependencies between logical components of a data processing environment, the pogram code comprising:

at least one monitoring agent for monitoring run-time activity data of logical components of the data processing environment, and for sending the monitored run-time activity data to a correlation identifier;

a correlation identifier for receiving, from the at least one monitor, monitored run-time activity data of each of a first logical component and a second logical component of a data processing environment, and for comparing the monitored run-time activity data of

the first logical component with the monitored run-time activity data of the second logical component to identify positive correlations between the monitored run-time activity of the first and second logical components; and

a dependency generator for responding to identification of a positive correlation between the monitored run-time activity of the first and second logical components by recording the existence of a dependency relationship between the first and second logical components.

24. A computer program product comprising program code recorded on a recording medium, for controlling the operation of a data processing system on which the program code executes to determine run-time dependencies between logical components of a data processing environment, the program code comprising:

a correlation identifier for receiving, from at least one monitoring agent, monitored run-time activity data of each of a first logical component and a second logical component of a data processing environment, and for comparing the monitored run-time activity data of the first logical component with the monitored run-time activity data of the second logical component to identify positive correlations between the monitored run-time activity of the first and second logical components; and

a dependency generator for responding to identification of a positive correlation between the monitored run-time activity of the first and second logical components by generating a representation of the existence of a dependency relationship between the first and second logical components.

25. A data processing apparatus comprising:

a data processing unit;

a data storage unit;

a correlation identifier for receiving, from at least one monitoring agent, monitored run-time activity data of each of a first logical component and a second logical component of a data processing environment, and for comparing the monitored run-time activity data of the first logical component with the monitored run-time activity data of the second logical component to identify positive correlations between the monitored run-

time activity of the first and second logical components; and

a dependency generator for responding to identification of a positive correlation between the monitored run-time activity of the first and second logical components by recording in the data storage unit the existence of a dependency relationship between the first and second logical components.

26. The data processing apparatus of claim 25, wherein the dependency generator is configured to generate a CIM_dependency class instance to represent an identified positive correlation, the apparatus further comprising a CIM object manager configured to record dependency information within a CIM repository in the data storage unit.

27. A distributed data processing system comprising:

a first data processing apparatus comprising a set of logical components to be monitored, and at least one monitoring agent for monitoring run-time activity data for the set of logical components and for sending the monitored run-time activity data to a correlation identifier on a second data processing apparatus; and

a second data processing apparatus comprising:

a data processing unit;

a data storage unit;

a correlation identifier for receiving, from the at least one monitoring agent, monitored run-time activity data of each of a first logical component and a second logical component of a data processing environment, and for comparing the monitored run-time activity data of the first logical component with the monitored run-time activity data of the second logical component to identify positive correlations between the monitored run-time activity of the first and second logical components; and

a dependency generator for responding to identification of a positive correlation between the monitored run-time activity of the first and second logical components by recording in the data storage unit the existence of a dependency relationship between the first and second logical components.

28. The distributed data processing system of claim 27, further comprising at least one management application program for analysing recorded dependency relationships.

29. The distributed data processing system of claim 27, wherein the at least one management application program comprises a fault management application program.

30. A method for discovering dependencies between monitored components of a managed data processing system, comprising the steps of:

accessing, from the managed system, run-time activity data for the monitored components;

comparing the accessed run-time activity data of the monitored components to identify positive correlations between the run-time activity of the monitored components; and

generating an identification of a dependency relationship between components for which a positive correlation is identified.

31. The method of claim 30, wherein the step of accessing comprises accessing the run-time activity data via an API provided by the managed system.

32. The method of claim 30, wherein the step of comparing the accessed run-time activity data comprises comparing a plurality of run-time activity metrics for each monitored component.

33. The method of claim 32, further comprising:

computing a value representing the consistency of identification of a positive correlation between components for the plurality of run-time activity metrics; and

storing the computed value in association with the generated identification of a dependency relationship.

34. The method of claim 32, wherein the step of comparing the accessed run-time activity data comprises comparing run-time activity data for a plurality of executions of

the monitored components, and wherein the method further comprises:

computing a single value representing the consistency of identification of a positive correlation between components, for the plurality of executions of the components and the plurality of run-time activity metrics; and

storing the computed single value in association with the generated identification of a dependency relationship.

35. The method of claim 30, wherein the step of comparing the accessed run-time activity data comprises comparing run-time activity data for a plurality of executions of the monitored components, and wherein the method further comprises:

computing a value representing the consistency of identification of a positive correlation between components for the plurality of executions of the components; and

storing the computed value in association with the generated identification of a dependency relationship.